PCT ORGANISATION MONDIALE DE LA PROPRIETE INTELLECTUELLE
Bureau international

DEMANDE INTERNATIONALE PUBLIEE EN VERTU DU TRAITE DE COOPERATION EN MATIERE DE BREVETS (PCT)

(51) Classification internationale des brevets 5 : G06F 9/46 — A1

(11) Numéro de publication internationale: **WO 94/14116**

(43) Date de publication internationale: 23 juin 1994 (23.06.94)

(21) Numéro de la demande internationale: PCT/FR93/01219

(22) Date de dépôt international: 9 décembre 1993 (09.12.93)

(30) Données relatives à la priorité:
92/14971 11 décembre 1992 (11.12.92) FR

(71) Déposant *(pour tous les Etats désignés sauf US)*: BULL S.A. [FR/FR]; Tour Bull, 1, place Carpeaux, F-92800 Puteaux (FR).

(72) Inventeurs; et
(75) Inventeurs/Déposants *(US seulement)*: AYDIN, Alev [FR/FR]; 11, chemin des Vignes, F-78990 Elancourt (FR). BESNIER, Annick [FR/FR]; 4, domaine des Aulnes, F-78830 Bullion (FR). JACOBS, Margaret [FR/FR]; 11, rue des Prairies, F-78230 Le Pecq (FR).

(74) Mandataire: DEBAY, Yves; Bull S.A., PC: TB/2803, Tour Bull Cédex 74, F-92039 Paris-La Défense (FR).

(81) Etats désignés: CA, JP, US.

Publiée
Avec rapport de recherche internationale.
Avant l'expiration du délai prévu pour la modification des revendications, sera republiée si de telles modifications sont reçues.

(54) Title: DEVICE USING REMOTE PSEUDO-SOCKET FUNCTIONS

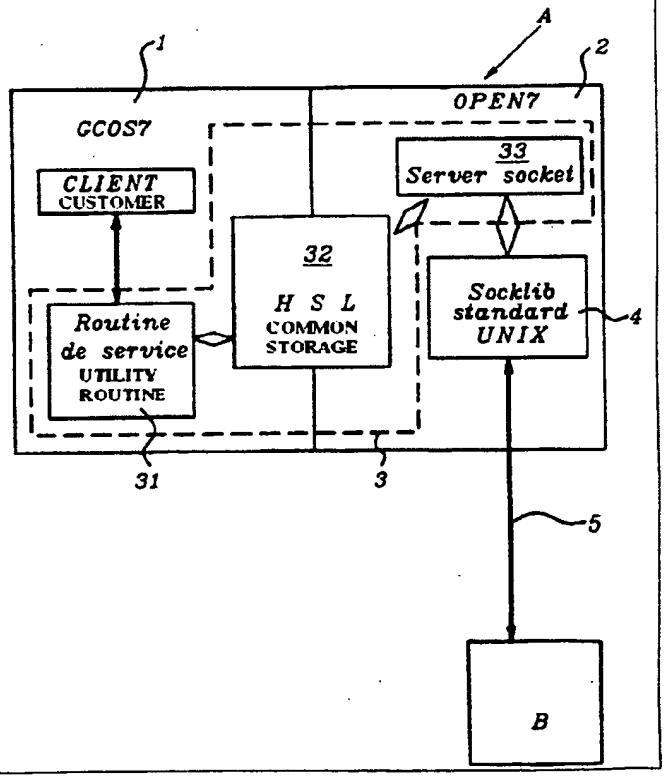
(54) Titre: DISPOSITIF D'UTILISATION DE FONCTIONS DE PSEUDO POINT DE COMMUNICATION DEPORTEES (PSEUDO-SOCKETS)

(57) Abstract

A device for using an open systems remote procedure call using the socket method in a proprietary application using primitives absent from the proprietary system and operating on a computer system provided with an open sub-system, using the socket function comprising: communication means between the proprietary application and an application under the open sub-system; coding means for coding the proprietary application primitives in a special format and storing said primitives; means for running an application under the open system ("UNIX") to decode and execute the function requested by the primitive and send back the result; means for decoding the result; and means providing common storage access synchronisation.

(57) Abrégé

La présente invention concerne un dispositif permettant d'utiliser une procédure d'appel à distance de systèmes ouverts utilisants le procédé point de communication dans une application propriétaire utilisant des primitives inexistantes dans le système propriétaire et fonctionnant sur un système informatique disposant d'un sous-système ouvert, utilisant la fonction point de communication comportant: des moyens de communiquer entre l'application propriétaire et une application sous le sous-système ouvert; des moyens de coder dans un format spécial les primitives de l'application propriétaire et de stocker ces primitives; des moyens de lancer une application sous le système ouvert ("UNIX") pour décoder, faire exécuter la fonction demandée par la primitive et de renvoyer le résultat; des moyens de décoder le résultat; des moyens d'assurer la synchronisation des accès à la mémoire partagée.



UNIQUEMENT A TITRE D'INFORMATION

Codes utilisés pour identifier les Etats parties au PCT, sur les pages de couverture des brochures publiant des demandes internationales en vertu du PCT.

AT	Autriche	**GB**	Royaume-Uni	**MR**	Mauritanie
AU	Australie	**GE**	Géorgie	**MW**	Malawi
BB	Barbade	**GN**	Guinée	**NE**	Niger
BE	Belgique	**GR**	Grèce	**NL**	Pays-Bas
BF	Burkina Faso	**HU**	Hongrie	**NO**	Norvège
BG	Bulgarie	**IE**	Irlande	**NZ**	Nouvelle-Zélande
BJ	Bénin	**IT**	Italie	**PL**	Pologne
BR	Brésil	**JP**	Japon	**PT**	Portugal
BY	Bélarus	**KE**	Kenya	**RO**	Roumanie
CA	Canada	**KG**	Kirghizistan	**RU**	Fédération de Russie
CF	République centrafricaine	**KP**	République populaire démocratique de Corée	**SD**	Soudan
CG	Congo			**SE**	Suède
CH	Suisse	**KR**	République de Corée	**SI**	Slovénie
CI	Côte d'Ivoire	**KZ**	Kazakhstan	**SK**	Slovaquie
CM	Cameroun	**LI**	Liechtenstein	**SN**	Sénégal
CN	Chine	**LK**	Sri Lanka	**TD**	Tchad
CS	Tchécoslovaquie	**LU**	Luxembourg	**TG**	Togo
CZ	République tchèque	**LV**	Lettonie	**TJ**	Tadjikistan
DE	Allemagne	**MC**	Monaco	**TT**	Trinité-et-Tobago
DK	Danemark	**MD**	République de Moldova	**UA**	Ukraine
ES	Espagne	**MG**	Madagascar	**US**	Etats-Unis d'Amérique
FI	Finlande	**ML**	Mali	**UZ**	Ouzbékistan
FR	France	**MN**	Mongolie	**VN**	Viet Nam
GA	Gabon				

DISPOSITIF D'UTILISATION DE FONCTIONS DE PSEUDO POINT DE COMMUNICATION DEPORTEES (PSEUDO-SOCKETS).

La présente invention concerne un dispositif d'utilisation de procédure d'appel à distance de fonctions de pseudo point de communication déportées (pseudo-sockets) et le procédé mis en oeuvre par le dispositif. Il est connu des systèmes informatiques comportant un système d'exploitation propriétaire. Il est également connu dans l'art antérieur des systèmes dits ouverts, par exemple de type "UNIX" qui compte tenu de leurs programmes et de leur constitution comportent des procédures d'appel à distance pour appeler d'autres systèmes ouverts implantés sur des stations éloignées. Il est également connu sur des systèmes informatiques pourvus de système d'exploitation propriétaire, c'est à dire spécifique au constructeur, d'implanter également sur ces systèmes un sous-système ouvert, par exemple du type "UNIX" qui offre les fonctions "prises de communication" (socket) permettant ainsi d'établir une communication à distance avec une autre station. Toutefois ces systèmes ne permettent pas aux applications tournant sur un système d'exploitation spécifique à un constructeur par exemple GCOS de communiquer avec des systèmes ouverts, par exemple du type "UNIX" et en particulier avec des stations éloignées.

Un premier but de l'invention est de proposer un dispositif permettant d'effectuer une liaison entre les applications tournant sur le système d'exploitation spécifique au constructeur pour communiquer, à travers le sous-système ouvert implanté sur la même machine, avec des stations éloignées.

Ce but est atteint par le fait que le dispositif, permettant d'utiliser une procédure d'appel à distance de systèmes ouverts utilisant le procédé point de communication (socket) dans une application propriétaire utilisant des primitives inexistantes dans le système d'exploitation propriétaire et fonctionnant sur un système informatique disposant d'un sous-système ouvert, ce sous-

système ouvert disposant de la fonction point de communication (socket), est caractérisé en ce qu'il comporte :

des moyens de communiquer entre l'application propriétaire et le sous-système ouvert via des segments de mémoire partagée ;

des moyens de coder dans un format spécial les primitives de l'application propriétaire qui n'existent pas dans le système d'exploitation propriétaire et de stocker ces primitives dans un segment de la mémoire partagée;

des moyens de lancer une application sur le système ouvert pour décoder, et faire exécuter la fonction demandée par la primitive et de renvoyer dans le même segment de mémoire le résultat de la fonction exprimée dans le format spécial;

des moyens de décoder le résultat de la fonction exprimée dans le format spécial de façon à ce que l'application propriétaire ait la visibilité "normale" de la fonction, c'est-à-dire comme si la fonction est exécutée localement par le système d'exploitation propriétaire ;

des moyens d'assurer la synchronisation des accès à la mémoire partagée.

Selon une autre particularité, les moyens d'assurer la synchronisation sont un premier, deuxième, et troisième sémaphores.

Selon une autre particularité, lorsque l'application propriétaire exécute une opération P sur le premier sémaphore, l'application du sous-système ouvert exécute une opération P sur le deuxième sémaphore, puis l'application propriétaire exécute une opération V sur le deuxième sémaphore et une opération P sur le troisième

sémaphore, ensuite l'application du sous-système ouvert exécute une opération V sur le troisième sémaphore et enfin l'application propriétaire exécute une opération V sur le premier sémaphore.

Selon une autre particularité, le format spécial dans lequel sont codées les primitives inexistantes dans le système d'exploitation propriétaire comprend une zone message constituée d'un premier champ formé par un entier représentant la fonction ;

un deuxième champ représentant le numéro de process pour lequel la demande est faite ;

un troisième champ indiquant la valeur de la fonction après l'exécution de la fonction ;

un quatrième champ indiquant par la valeur 0 que la fonction s'est exécutée normalement ;

un cinquième et un sixième champ pour identifier la machine qui fait la demande ;

un septième champ réservé ;

des huitième, neuvième, dixième, onzième, douzième, treizième champs destinés chacun à stocker des valeurs de paramètre représentées par un entier et;

un quatorzième champ tampon destiné à stocker des caractères représentant les valeurs de paramètre autre que des entiers.

Selon une autre particularité, les moyens de codage des primitives de l'application propriétaire vers le format spécial comportent une table de correspondance des primitives vers des valeurs entières qui représentent la fonction spécifique et un programme spécifique pour chaque

fonction permettant de remplir les champs utiles du format.

Selon une autre particularité, les moyens de décodage du format spécial vers les primitives du sous-système ouvert comportent une table de correspondance des valeurs entières vers les primitives du sous-système ouvert et un programme de traitement spécifique des champs du format utile à chaque primitive du sous-système ouvert.

Un autre but de l'invention est de proposer un procédé mis en oeuvre par le dispositif.

Ce but est atteint par le fait que le procédé d'appel à distance de stations éloignées à partir d'une station utilisant un système d'exploitation propriétaire et ayant un sous-système ouvert consiste dans les étapes suivantes :

- mise à l'état P d'un premier sémaphore par le système propriétaire et d'un deuxième sémaphore par le sous-système ouvert ;

- codage de la fonction à exécuter dans un format donné par une routine de service du système propriétaire ;

- écriture de cette fonction codée dans un segment de mémoire partagée ;

- mise à l'état V d'un deuxième sémaphore ;

- réveil par la mise à l'état V du deuxième sémaphore d'une application serveur de point de communication (socket serveur), fonctionnant dans le sous-système ouvert ;

- décodage par cette application du message contenu dans le segment et concomitamment mise à l'état P d'un

troisième sémaphore par le système d'exploitation propriétaire;

- exécution de la fonction décodée par la station éloignée;

- réception du résultat par le sous-système ouvert et recodage de ce résultat dans le format spécifique ;

- réécriture du résultat encodé dans le segment de mémoire partagée ;

- mise à l'état V du troisième sémaphore provoquant ainsi le réveil de l'application routine de service et mise à l'état P du deuxième sémaphore provoquant ainsi la mise en attente du socket serveur;

- interprétation par cette routine de service du résultat de la fonction ;

- mise à l'état V du premier sémaphore pour permettre à d'autres utilisateurs du système d'exploitation propriétaire d'effectuer un autre appel à distance de cette manière, sans qu'il y ait mélange d'informations par les deux utilisateurs.

D'autres particularités et avantages de la présente invention apparaitront plus clairement à la lecture de la description ci-après, faite en référence aux figures dans lesquelles :

La figure 1 représente une vue schématique du dispositif selon l'invention,

La figure 2 représente un organigramme du procédé mis en oeuvre par le dispositif,

La figure 3 représente le format de codage d'une fonction stockée dans une zone message de la mémoire partagée,

La figure 4 représente le tableau de correspondance utilisé par la routine de service ou par le serveur de point de communication.

La figure 1 représente un système informatique (A) comportant un système d'exploitation propriétaire, par exemple du type GCOS7, ce système informatique (A) pouvant également fonctionner avec un sous-système ouvert par exemple OPEN7 du type "UNIX". Ce sous-système ouvert "UNIX" (2) comporte une bibliothèque ("UNIX") standard (4) permettant d'utiliser les fonctions point de communication (socket) d'environnement "UNIX". Toutefois dans le système informatique de l'art antérieur, il n'existe aucune communication possible entre le système d'exploitation propriétaire et le sous-système ouvert permettant ainsi la mise en oeuvre de fonction de communication à distance du type point de communication (socket). Ces fonctions point de communication à distance permettent aux systèmes fonctionnant à l'aide du sous-système OPEN7 de communiquer via la ligne de communication (5) avec une autre machine (B) du type fonctionnant avec un système ouvert par exemple du type "UNIX".

L'invention concerne principalement le dispositif permettant au système d'exploitation propriétaire (1) de communiquer avec le sous-système (2) et par l'intermédiaire de ces fonctions point de communication du sous-système ouvert "UNIX" avec une station (B) éloignée de type "UNIX". Le dispositif comporte une routine de service de procédure d'appel éloignée (31) (run time remote procedure call). Cette routine (31) communique avec un segment de mémoire partagée (32) de la machine et à travers ce segment de mémoire partagée (32) avec un serveur de point de communication (33) (socket serveur) qui communique avec la bibliothèque standard (4) des

fonctions de communication de type du sous-système ouvert ("UNIX") permettant la communication avec une station éloignée. Le procédé mis en oeuvre par le dispositif et les moyens utilisés dans le dispositif va maintenant être décrit à l'aide des figures 2,3,4.

La routine de service de RPC (Remote Procedure Call) contient un point d'entrée correspondant à chaque fonction "UNIX" qui n'existe pas dans le système d'exploitation propriétaire. Les paramètres et les noms de tous ces points d'entrée sont identiques à ceux qui correspondent aux fonctions manquantes. Donc, automatiquement, lorsqu'une application propriétaire appelle une fonction manquante, c'est le point d'entrée correspondant de la routine de service qui est activé. Cette routine de service fonctionne comme explicité à la figure 2, en commençant par faire une opération P sur un sémaphore (gmutex) comme représenté à l'étape (311) puis ensuite à exécuter un codage représenté à l'étape (312), ce codage consistant à traduire dans un format représenté à la figure 3 la fonction non existante pour le système d'exploitation propriétaire en un format déterminé. Ensuite l'étape (313) consiste à écrire cette fonction codée dans le format déterminé dans un segment de mémoire (321) d'une mémoire partagée (32). La routine de service (31) effectue ensuite comme représenté à l'étape (314) une opération V sur un sémaphore (mutexur) et à l'étape (315) une opération P sur un sémaphore (mutexgr) puis se met en attente comme représenté à l'étape (316). L'exécution de l'opération V sur le sémaphore (mutexur) déclenche le réveil du serveur de point de communication (33) qui avait été mis en attente lors de l'étape (331) par une opération P sur le sémaphore (mutexur) dans l'état d'attente représenté à l'étape (332). L'opération V sur le sémaphore (mutexur) exécutée par la routine de service RPC déclenche donc le réveil du serveur de point de communication et provoque l'étape (333) de fin d'attente qui ensuite lance la procédure de décodage (334) du message lu dans la zone

(321) de la mémoire partagée. Cette étape de décodage est ensuite suivie d'une étape d'exécution (335) de la fonction. Cette étape d'exécution s'effectuera sur une station éloignée par exemple du type "UNIX" B après avoir mis en oeuvre le mécanisme de point de communication standard existant dans le sous-système ouvert par exemple "UNIX" par utilisation de la bibliothèque de fonctions de communication standard (4). Cette étape d'exécution est suivie d'une étape de codage (336) du résultat de la fonction, ce résultat étant reçu par la partie sous-système ouvert par exemple "UNIX" de la station A et provenant de la station B. Ce codage s'effectue selon le format représenté à la figure 3 et à l'étape (337) le serveur (33) écrit le résultat dans le segment (321) de la mémoire partagée. Après cette étape, le serveur exécute une opération V sur le sémaphore (mutexgr), cette opération V ayant pour but de réveiller la routine de service RPC en provoquant l'étape de fin d'attente (317) qui lance ensuite l'étape (318) d'interprétation du résultat de la fonction par le système d'exploitation propriétaire et ensuite une étape d'opération V sur le sémaphore (gmutex). Cette opération V ayant pour but de permettre à d'autres utilisateurs d'effectuer un travail similaire. L'opération P à l'étape (311) a pour but d'éviter que d'autres utilisateurs du système d'exploitation écrivent dans le segment partagé en même temps, et l'opération P à l'étape (331) a pour but de mettre le serveur (33) de point de communication en attente de la fin d'écriture dans le segment de mémoire partagée. Avant toute utilisation de la routine de service RPC (31) sur le système d'exploitation propriétaire, il faut au préalable lancer dans le sous-système ouvert par exemple "UNIX" le serveur de point de communication (33) pour qu'il exécute l'étape (330) qui consiste à créer un segment de mémoire partagée (32) et à demander l'allocation de trois sémaphores (mutexur, gmutex et mutexgr) et ensuite à se mettre en attente sur le sémaphore (mutexur). La fonction codée par la routine de

service comprendra les informations représentées à la figure 3 et ces informations seront stockées dans une zone de message au format représenté à la figure 3. Cette zone de message comprend un premier champ dont la valeur représente la fonction (func_no), un deuxième champ qui représente le numéro J et le numéro P pour lesquels la demande est faite (JP), un troisième champ (func-value) indiquant la valeur de la fonction après l'exécution de la fonction. Cette valeur étant remplie par le serveur de point de communication (33) à l'étape (337). Un quatrième champ (loc-errno) qui indique par la valeur 0 que la fonction s'est exécutée normalement ;

un cinquième champ (sys_ser_num) et un sixième champ (bksto) qui identifient la machine qui fait la demande ;

un septième champ réservé (RFU1) ;

des huitième (INT1), neuvième (INT2), dixième (INT3), onzième (INT4), douzième (INT5), treizième (INT6) champs destinés chacun à stocker une valeur de paramètre représentée par un entier ;

et enfin un quatorzième champ (BUF) destiné à constituer un tampon pour stocker des caractères représentant les valeurs des paramètres autres que des entiers.

La valeur représentative d'une fonction inscrite dans le premier champ est traduite au codage par la routine de service (31) ou au décodage par le serveur de point de communication (33) selon le tableau figurant à la figure 4. Ce tableau représente les fonctions principales mises en oeuvre dans le dispositif de l'invention.

Nous rappellerons qu'une opération P fait une opération de décrémentation (-1) sur la valeur du sémaphore, et si la valeur ainsi obtenue est négative, le process est bloqué et attend le déblocage par une opération V qui par une

incrémentation (+1) rend positive la valeur du sémaphore. Les trois sémaphores sont créés en utilisant une fonction (ssemall), ce qui permet d'associer une adresse donnée à un sémaphore de type O, cette adresse étant communiquée au serveur (33). Le serveur écrit ensuite dans la zone d'entête de la mémoire partagée (header) les adresses des trois sémaphores et le système d'exploitation s'attache ensuite le segment de mémoire par une fonction (shmemat) ce qui lui permet ensuite de récupérer les adresses des trois sémaphores pour faire une opération V ou une opération P selon le choix du programme. Une telle opération s'effectue par la fonction (ssemop) qui selon le paramètre qu'on lui affecte permet la sélection entre l'opération V ou l'opération P.

Pour bien permettre la compréhension de l'invention, nous allons maintenant effectuer la description du codage et décodage par la routine de service (31) et le serveur (33), pour quelques exemples de fonctions qui figurent dans la table de la figure 4.

Exemple 1 :

Chaque fois qu'un client GCOS7 demande une fonction inexistante, par exemple la fonction "socket", la routine de service (31) effectue le codage de cette fonction en remplissant la zone message comme explicité ci-dessous :

func_no = 3
int1 = dom
int2 = type
int3 = prot.

les champs jp, sys_ser-num et bkst0 sont également remplis.

Le message après écriture dans le segment de mémoire partagée est ensuite décodé par le serveur (33) qui sait

par programme que pour la fonction "socket" (func_no = 3) seuls les champs ci-dessus sont valables en entrée.

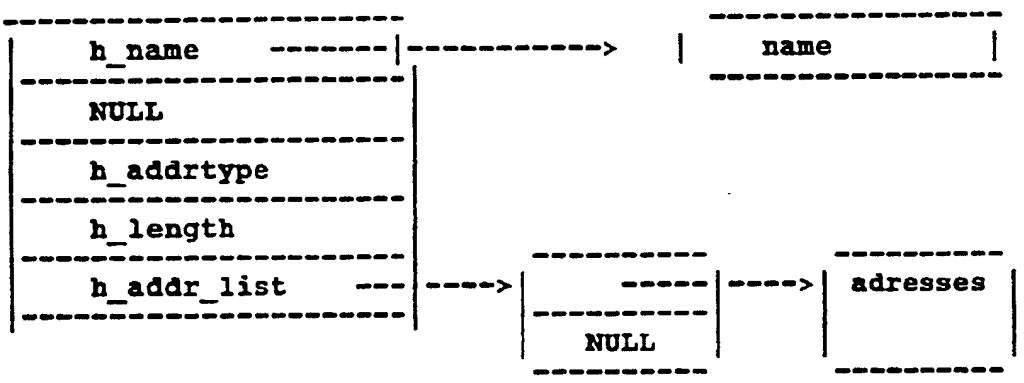
Le serveur exécute ensuite la fonction soit en local, soit en interaction avec une station éloignée et code le résultat de la façon suivante :

func_value = socket (int1, int2, int3)
si func_value = -1, alors le champ loc_errno = -1.

La réécriture s'effectue uniquement dans ces champs du segment et pour son étape d'interprétation la routine de service (31) sait que seuls les champs func_value et loc_errno sont valables au retour de la fonction "socket".

Exemple 2 :
La fonction "gethostbyname" permet à un utilisateur "UNIX", ne sachant que le nom du site "host", d'obtenir d'autres renseignements utiles concernant le site "host". Tous ces renseignements sont regroupés dans des structures (comme indiqué par notre schéma). Les flèches ne représentent pas des correspondances mais sont des pointeurs d'une structure vers une autre structure ou vers une chaîne de caractères (par exemple, "h_name"). De cette façon, il suffit de connaître l'adresse de la première structure pour accéder à l'ensemble des informations utiles concernant le site "host". C'est pour cette raison que la valeur de la fonction est l'adresse de la première structure.

Lorsque la fonction demandée par GCOS7 est la fonction "*gethostbyname (name)", le programme de codage (31)sait que cette fonction a un seul paramètre d'entrée, constitué par une chaîne de caractères qui représente le nom de la machine (host). La valeur de la fonction est l'adresse d'une structure complexe, certains champs de la structure servant de pointeurs vers d'autres structures ou chaînes de caractères.



Pour passer cette demande de fonction au sous-système OPEN7 d'"UNIX", la zone message du segment de mémoire est remplie comme ci-dessous :

func_no = 7
buf = name.

Puis, le codage effectue un transcodage EBCDIC en code ASCII sur le contenu (du buffer) du tampon, car du côté "UNIX", seuls les caractères ASCII sont attendus.
Les champs jp, sys_ser_num et bkst0 sont également remplis.

Le message est ensuite écrit dans le segment de mémoire et est ensuite décodé par le serveur (33), qui sait que pour la fonction "gethostbyname", seul le champ tampon est valable en entrée.

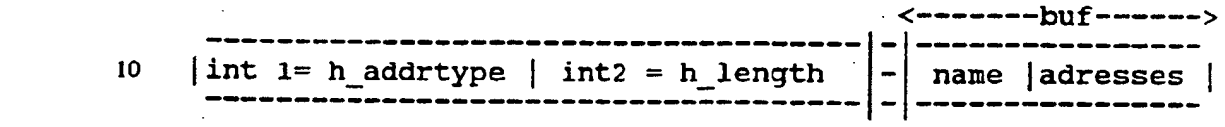
L'exécution s'effectue de la façon suivante :

structure_adresse = gethostbyname (buffer)
si structure_adresse = NULL, alors loc_errno = -1.

Comme cela n'a pas de sens de renvoyer au système d'exploitation propriétaire l'adresse d'une structure qui se trouve sur le sous-système "UNIX", nous effectuons un décodage du résultat de la fonction pour transmettre au système d'exploitation propriétaire toutes les

informations pointées par l'adresse rendue. Ceci est fait de la façon suivante :

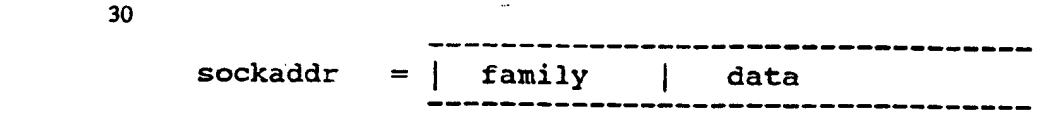
le champ tampon (buf) est rempli avec le nom (name) et ensuite les adresses pointées indirectement par h_addr_list, informations qui sont concaténées au nom (name), qui a une longueur fixe de 14 caractères.



Le message de retour est renvoyé au système d'exploitation propriétaire via le segment de mémoire partagée. Au cours de l'interprétation par GCOS7, on sait que les champs (int1), (int2) et (buf) doivent être utilisés pour allouer et remplir les structures retournées par la fonction "gethostbyname".

Exemple 3 :

Dans le cas où la fonction inexistante est la fonction "sendto", la routine de service (31) sait que cette fonction a quatre paramètres de type entier (s, len, flags et fromlg). i = sendto (s, buffer, len, flags, from, fromlg). Le paramètre "buffer" est une chaîne de caractères et "from" est un pointeur vers une structure sockaddr du format suivant :



ou "family" est un demi-mot de format entier, "data" est une chaîne de caractères de longueur 14.

Tous les paramètres sont des paramètres d'entrée.

On utilise le terme "paramètre d'entrée" pour distinguer un paramètre qui sert à fournir des informations à une

fonction (permettant ainsi à la fonction de s'exécuter correctement) d'un paramètre de sortie qui contient des informations rendues par la fonction exécutée. Certains paramètres peuvent être comme dans la fonction "select", des paramètres d'entrée et de sortie.

Pour passer cette demande de fonction au serveur (33), la zone de message est remplie comme explicité ci-dessous :

func_no = 11 (la fonction sendto)
int1 = s
int2 = len
int3 = flags
int4 = fromlg

Le champ tampon (buf) est rempli avec buffer et ensuite la structure pointée par from est concaténée à buffer.

| buffer | sockaddr |

Les champs jp, sys_ser_num et bkst0 sont également remplis.

Le message est écrit dans le segment de mémoire partagée et il est ensuite décodé par le serveur (33) qui sait que pour la fonction sendto, seuls les champs ci-dessus sont valables en entrée.

Le serveur (33) exécute la fonction de la façon suivante :

le champ tampon (buf) est d'abord décodé pour récupérer les valeurs de buffer et de from_structure.

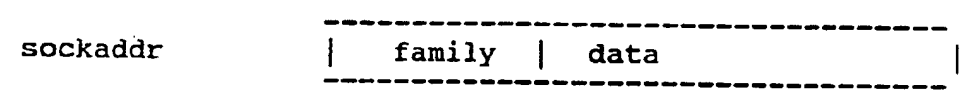
func_value = sendto (int1, buffer, int2, int3, &sockaddr, int4), &sockaddr représentant l'adresse de sockaddr, le résultat de l'exécution se présentant sous la forme :

si func_value = -1, alors loc_errno = -1.

Le message de retour est envoyé au système d'exploitation propriétaire via le segment de mémoire partagée. Au retour dans le système d'exploitation, on sait que seuls les champs func_value et loc_errno sont valables pour la fonction "sendto".

Exemple 4 :

Lorsque la fonction appelée par le client GCOS7 est "recvfrom", la routine de service sait que cette fonction a trois paramètres de type entier (s, len et flags). i= recvfrom (s, buffer, len, flags, from, fromlg). Le paramètre "buffer" est une chaîne de caractères et "from" est un pointeur vers une structure du format suivant :



ou "family" est un demi-mot de format entier, "data" est une chaîne de caractères de longueur 14.

le paramètre "fromlg" pointe vers une donnée en format entier.

Les paramètres s, len et flags sont des paramètres d'entrée. "fromlg" est un paramètre d'entrée et de sortie, alors que "buf" et "from" sont des paramètres de sortie uniquement.

Pour passer cette demande de fonction au serveur (33), la zone de message est remplie comme explicité ci-dessous :

func_no = 10 (la fonction recvfrom)
int1 = s
int2 = len
int3 = flags

int4 = *fromlg (int4 = la valeur de l'entier pointée par fromlg)

Le champ tampon (buf) sera utilisé pour contenir, au retour de OPEN7, les informations suivantes :

buffer	sockaddr

Les champs jp, sys_ser_num et bkst0 sont également remplis.

Le message est écrit dans le segment de mémoire partagée et il est ensuite décodé par le serveur (33) qui sait que pour la fonction "recvfrom", seuls les champs ci-dessus sont valables en entrée.

Le serveur (33) effectue la fonction de la façon suivante :

func_value = recvfrom (int1, buffer, int2, int3, &from_copy, &int4), &from_copy représentant l'adresse de from_copy, &int4 représentant l'adresse de int4.

Le résultat de l'exécution se présentent sous la forme :

si func_value = -1, alors loc_errno = -1.

Le contenu du tampon (buf) est rempli directement par l'appel de la fonction "recvfrom". Le contenu de from (pointé côté OPEN7 par l'adresse de from_copy) est ensuite concaténé comme décrit ci-dessus.

Le message de retour est renvoyé au système d'exploitation propriétaire via le segment de mémoire partagée. Au retour dans le système d'exploitation, on sait que seuls les champs (func_value), (loc_errno), (int4) et (buf) sont valables pour la fonction "recvfrom".

Exemple 5 :

Enfin lorsque la fonction appelée par le client est la fonction "select", l'opération de codage effectuée par la routine de service (31) sait que cette fonction a cinq paramètres.

```
i = select  (nfds, readfds, writefds, execpfds, timeout)
                |       |        |         |        |
              entier pointeur pointeur   pointeur
```

le paramètre "nfds" est un paramètre d'entrée et les autres sont des paramètres d'entrée et de sortie, ces quatre derniers sont des pointeurs vers des structures de taille fixe.

Pour passer cette demande de fonction au serveur (33), la zone de message est remplie comme explicité ci-dessous :

func_no = 13 (la fonction select)
int1 = nfds.

Le champ tampon (buf) est ensuite rempli de la façon suivante :

structure pointée par readfds	structure pointée par writefds	structure pointée par execpfds	structure pointée par timeout

Les paramètres readfds, writefds, execpfds et timeout pouvant être nuls, les champs correspondant dans le tampon (buf) seront omis dans ce cas. Ceci est signalé à OPEN7 par le positionnement à 1 ou à 0 des champs int2, int3, int4, int5 selon la présence ou non du paramètre dans buf.

int2 = 0 ---> readfds est NULL
int2 = 1 ---> readfds est non-NULL

int3 = 0 ---> writefds est NULL
int3 = 1 ---> writefds est non-NULL
int4 = 0 ---> execpfds est NULL
int4 = 1 ---> execpfds est non-NULL
int5 = 0 ---> timeout est NULL
int5 = 1 ---> timeout est non-NULL

Les champs jp, sys_ser_num et bkst0 sont également remplis.

Le message est écrit dans le segment de mémoire partagée et il est ensuite décodé par le serveur (33) qui sait que pour la fonction "select", seuls les champs ci-dessus sont valables en entrée.

Le serveur (33) effectue la fonction de la façon suivante :

func_value = select (int1, readptr, writeptr, execpptr, timeoutptr),

readptr, writeptr, execpptr et timeoutptr sont des pointeurs vers des structures OPEN7 du même format que celles pointées par readfds, writefds, execpfds et timeout.

Le résultat de l'exécution se présentant sous la forme :

si func_value = -1, alors loc_errno = -1.

Le contenu des structures pointées par readptr, writeptr, execpptr et timeoutptr est ensuite inséré dans le tampon (buf) selon les valeurs de (int2), (int3), (int4) et (int5).

Le message de retour est renvoyé au système d'exploitation propriétaire via le segment de mémoire partagée. Au retour

dans le système d'exploitation, on sait quels sont les champs valables pour la fonction "select".

Les exemples de fonctions données ci-dessus sont purement explicatifs et non limitatifs et toutes modifications à la portée de l'homme de métier fait également partie de l'esprit de l'invention.

REVENDICATIONS

1.- Dispositif permettant d'utiliser une procédure d'appel à distance (REMOTE PROCEDURE CALL) de systèmes ouverts utilisant le procédé point de communication (socket) dans une application propriétaire (proprietary) utilisant des primitives inexistantes dans le système d'exploitation propriétaire et fonctionnant sur un système informatique disposant d'un sous-système ouvert ("UNIX"), ce sous-système ouvert disposant de la fonction point de communication (socket) caractérisé en ce qu'il comporte:

des moyens de communiquer entre l'application propriétaire et une application sous le sous-système ouvert via des segments de mémoire partagée;

des moyens de coder dans un format spécial les primitives de l'application propriétaire qui n'existent pas dans le système d'exploitation propriétaire et de stocker ces primitives dans un segment de la mémoire partagée;

des moyens de lancer une application sous le système ouvert ("UNIX") pour décoder, faire exécuter la fonction demandée par la primitive et de renvoyer dans le même segment de mémoire le résultat de la fonction exprimée dans le format spécial;

des moyens de décoder le résultat de la fonction exprimée dans le format spécial pour donner à l'application propriétaire une même visibilité comme si la fonction était effectuée localement par le système d'exploitation propriétaire;

des moyens d'assurer la synchronisation des accès à la mémoire partagée.

2. Dispositif selon la revendication 1 caractérisé en ce que les moyens d'assurer la synchronisation sont un premier (gmutex), deuxième (mutexur) et troisième (mutexgr) sémaphores.

3. Dispositif selon la revendication 2 caractérisé en ce que lorsque l'application propriétaire exécute une opération P sur le premier sémaphore (gmutex), l'application du sous-système ouvert exécute une opération P sur le deuxième sémaphore (mutexur), puis l'application propriétaire exécute une opération V sur le deuxième sémaphore (mutexur) et une opération P sur le troisième sémaphore (mutexgr), ensuite l'application exécute une opération V sur le troisième sémaphore (mutexgr) et enfin l'application propriétaire exécute une opération V sur le premier sémaphore (gmutex).

4. Dispositif selon une des revendications précédentes caractérisé en ce que le format spécial dans lequel sont codées les primitives (RPC) inexistantes dans le système d'exploitation propriétaire comprend une zone message constituée:

d'un premier champ (func-n°) formé par un entier représentant la fonction;

d'un deuxième champ (JP) représentant le J n° et le P n° pour lequel la demande est faite;

d'un troisième champ (func-value) indiquant la valeur de la fonction après l'exécution de la fonction;

d'un quatrième champ (loc-errn°) indiquant par la valeur zéro que la fonction s'est exécutée normalement;

d'un cinquième (sys-ser-num) et sixième (bksto) champ pour identifier la machine qui fait la demande;

d'un septième champ réservé;

de huitième (int1), neuvième (int2), dixième (int3), onzième (int4), douzième (int5), treizième (int6) champs destinés chacun à stocker une valeur de paramètre représentée par un entier;

d'un quatorzième champ tampon (buf) destiné à stocker des caractères représentant les valeurs des paramètres autres que des entiers.

5. Dispositif selon la revendication 4 caractérisé en ce que les moyens de codage des primitives de l'application propriétaire vers le format spécial comportent une table de correspondance des primitives vers des valeurs entières qui représentent la fonction spécifique et vers un programme spécifique pour chaque fonction permettant de remplir de façon adéquate les champs utiles du format.

6. Dispositif selon la revendication 4 caractérisé en ce que les moyens de décodage du format spécial vers des primitives du sous-système ouvert comportent une table de correspondance des valeurs entières vers des primitives du sous-système ouvert et vers un programme de traitement spécifique des champs du format utiles à chaque primitive du sous-système ouvert.

7. Procédé mis en oeuvre par le dispositif selon l'une des revendications précédentes, caractérisé en ce que le procédé d'appel à distance de stations éloignées à partir d'une station utilisant un système d'exploitation propriétaire et ayant un sous-système ouvert consiste dans les étapes suivantes :

- mise à l'état P d'un premier sémaphore par le système propriétaire et d'un deuxième sémaphore par le sous-système ouvert ;

- codage de la fonction à exécuter dans un format donné par une routine de service du système propriétaire ;

- écriture de cette fonction codée dans un segment de mémoire partagée ;

- mise à l'état V d'un deuxième sémaphore ;

- réveil par la mise à l'état V du deuxième sémaphore d'une application serveur de point de communication (socket serveur), fonctionnant dans le sous-système ouvert ;

- décodage par cette application du message contenu dans le segment et concomitamment mis à l'état P d'un troisième sémaphore par le système d'exploitation propriétaire ;

- exécution de la fonction décodée par la station éloignée;

- réception du résultat par le sous-système ouvert et recodage de ce résultat dans le format spécifique ;

- réécriture du résultat encodé dans le segment de mémoire partagée ;

- mise à l'état V du troisième sémaphore provoquant ainsi le réveil de l'application routine de service et mise à l'état P du deuxième sémaphore provoquant ainsi la mise en attente du socket serveur ;

- interprétation par cette routine de service du résultat de la fonction ;

- mise à l'état V du premier sémaphore pour permettre à d'autres utilisateurs du système d'exploitation propriétaire d'effectuer un autre travail.

8. Procédé selon la revendication 7 caractérisé en ce que avant toute utilisation de la routine de service (31) pour l'exécution d'une fonction inexistante sur le système d'exploitation propriétaire, il existe une étape de lancement du serveur de point de communication (33) qui dans une première étape crée un segment de mémoire partagée (32) et demande l'allocation des trois sémaphores (gmutex, mutexur et mutexgr), avant de se mettre en attente sur "mutexur".

1/4

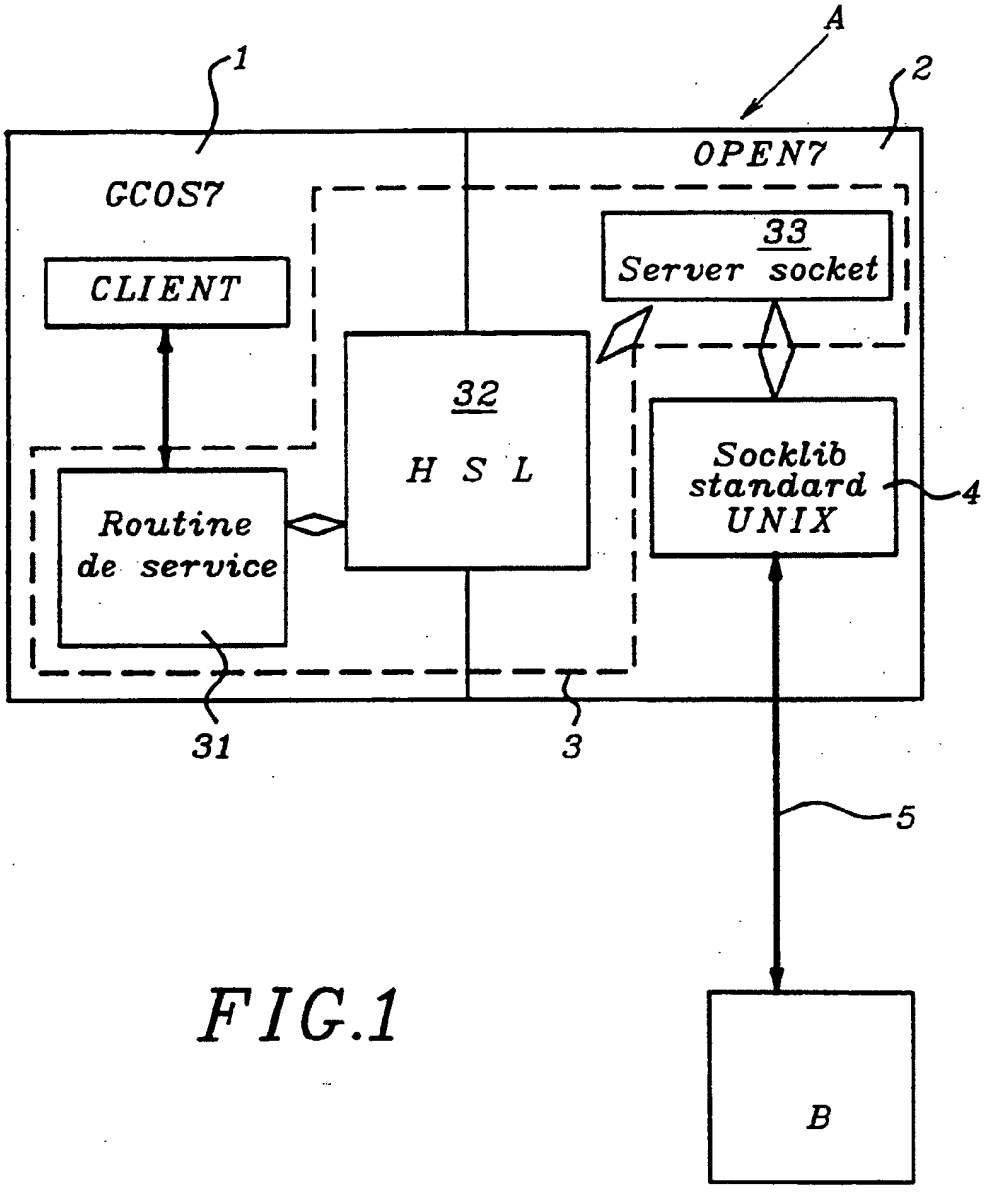


FIG.1

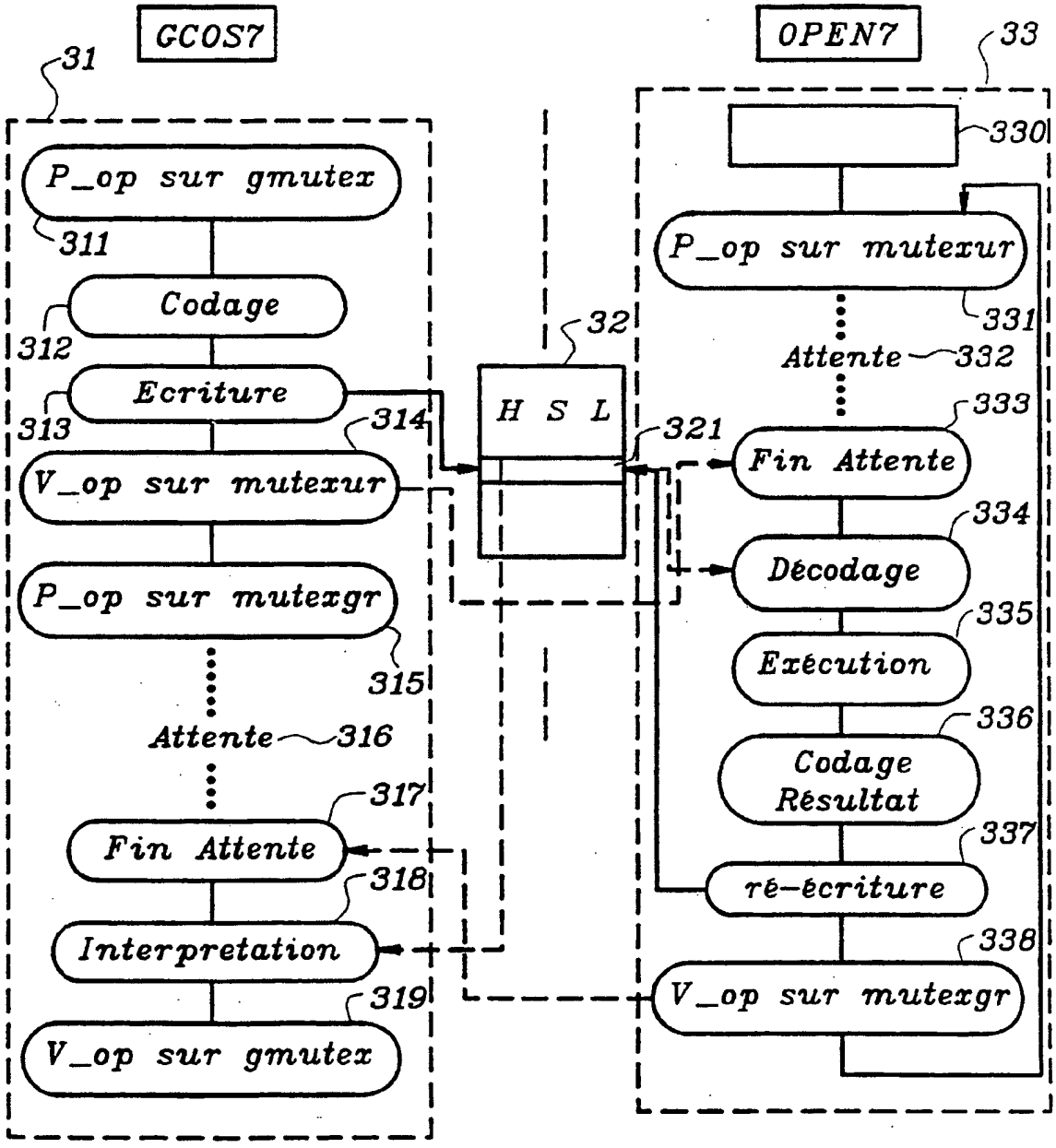


FIG.2

FIG.3

func_no (integer)	:	*une valeur qui represente la fonction*
jp (integer)	:	*le J-no et le P-no pour lesquels la demande est faite*
func_value (integer)	:	*la valeur de la fonction (remplie par socket_serv après l'execution de la fonction*
loc_errno (integer)	:	*0 si la fonction s'est executée normalement*
sys_ser_num (integer)	:	*pour identifier la machine qui fait la demande*
bkst0 (short)	:	*pour identifier la machine qui fait la demande*
rful (short)	:	*reserve*
int1,int2,int3, int4,int5,int6 (integer)	:	*ces champs servent a stocker les valeurs des parametres dans le cas de parametres "integer"*
buf (char[2000])	:	*une zone de caracteres permettant de stocker les valeurs de parametres de type autre que "integer"*

321,322

func_no	*JP*	*func_value*	*loc_errno*	*sys_ser_num*	*bkst0*	*rful*	*int1*	*int2*
int3	*int4*	*int5*	*int6*	*buf*				

Function name	Fonction number
uuid_gen	1
fcntl	2
socket	3
bind	4
close	5
gethostbyaddr	6
gethostbyname	7
gethostname	8
getsockname	9
recvfrom	10
sendto	11
setsockopt	12
select	13
closeall	14
gettimeofday	15
recvmsg	16
sendmsg	17
accept	18
connect	19
ioctl	20
listen	21
getsockopt	22
getpeername	23
send	24
recv	25

FIG.4

INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/FR 93/01219

A. CLASSIFICATION OF SUBJECT MATTER
IPC 5 G06F9/46

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 5 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP,A,0 371 229 (HEWLETT-PACKARD COMPANY) 6 June 1990 see abstract see figures 1,5 see column 1, line 24 - column 5, line 11 see claims 1-8	1,4-6
A	GB,A,2 176 636 (INTERNATIONAL STANDARD ELECTRIC CORPORATION) 31 December 1986 see page 1, left column, line 1 - right column, line 115; figures 1,2	1,4-6
A	US,A,5 123 094 (MACDOUGALL) 16 June 1992 see abstract; claim 1	1

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
5 April 1994	18/04/94

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+ 31-70) 340-2040, Tx. 31 651 epo nl, Fax (+ 31-70) 340-3016	Wiltink, J

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inter..onal Application No
PCT/FR 93/01219

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	UNIX REVIEW June 1987 , US pages 66 - 75 HERRICK J. JOHNSON: 'Each piece in its place' see the whole document	1,4-6
A	PROCEEDINGS OF THE 1989 INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING vol. II , 8 August 1989 , ST. CHARLES, IL, US, pages II-160 - II-169 XP78485 UMAKISHORE RAMACHANDRAN ET AL.: 'Coherence of Distributed Shared Memory: Unifying Synchronization and Data Transfer' see page II-164, left column, line 32 - right column, line 60	2,3,7,8
A	MAURICE J. BACH: 'The design of the UNIX operating system' , PRENTICE/HALL INTERNATIONAL, INC. , ENGLEWOOD CLIFFS, NJ 07632, US Chapitre 11: 'Interprocess communication', pages 355 - 390 see page 367, line 16 - page 389, line 6	1-8

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inter. onal Application No
PCT/FR 93/01219

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
EP-A-0371229	06-06-90	US-A-	5124909	23-06-92
		JP-A-	2171951	03-07-90
GB-A-2176636	31-12-86	NONE		
US-A-5123094	16-06-92	NONE		

RAPPORT DE RECHERCHE INTERNATIONALE

Den. e Internationale No
PCT/FR 93/01219

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
CIB 5 G06F9/46

Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)
CIB 5 G06F

Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche

Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si cela est réalisable, termes de recherche utilisés)

C. DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie *	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
X	EP,A,0 371 229 (HEWLETT-PACKARD COMPANY) 6 Juin 1990 voir abrégé voir figures 1,5 voir colonne 1, ligne 24 - colonne 5, ligne 11 voir revendications 1-8	1,4-6
A	GB,A,2 176 636 (INTERNATIONAL STANDARD ELECTRIC CORPORATION) 31 Décembre 1986 voir page 1, colonne de gauche, ligne 1 - colonne de droite, ligne 115; figures 1,2	1,4-6
A	US,A,5 123 094 (MACDOUGALL) 16 Juin 1992 voir abrégé; revendication 1	1

-/--

[X] Voir la suite du cadre C pour la fin de la liste des documents

[X] Les documents de familles de brevets sont indiqués en annexe

* Catégories spéciales de documents cités:

"A" document définissant l'état général de la technique, non considéré comme particulièrement pertinent
"E" document antérieur, mais publié à la date de dépôt international ou après cette date
"L" document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée)
"O" document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens
"P" document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée

"T" document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention
"X" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément
"Y" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier
"&" document qui fait partie de la même famille de brevets

Date à laquelle la recherche internationale a été effectivement achevée	Date d'expédition du présent rapport de recherche internationale
5 Avril 1994	18.04.94
Nom et adresse postale de l'administration chargée de la recherche internationale Office Européen des Brevets, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	**Fonctionnaire autorisé** Wiltink, J

Formulaire PCT/ISA/210 (deuxième feuille) (juillet 1992)

C.(suite) DOCUMENTS CONSIDERES COMME PERTINENTS

Categorie *	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
A	UNIX REVIEW Juin 1987 , US pages 66 - 75 HERRICK J. JOHNSON: 'Each piece in its place' voir le document en entier	1,4-6
A	PROCEEDINGS OF THE 1989 INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING vol. II , 8 Août 1989 , ST. CHARLES, IL, US, pages II-160 - II-169 XP78485 UMAKISHORE RAMACHANDRAN ET AL.: 'Coherence of Distributed Shared Memory: Unifying Synchronization and Data Transfer' voir page II-164, colonne de gauche, ligne 32 - colonne de droite, ligne 60	2,3,7,8
A	MAURICE J. BACH: 'The design of the UNIX operating system' , PRENTICE/HALL INTERNATIONAL, INC. , ENGLEWOOD CLIFFS, NJ 07632, US Chapitre 11: 'Interprocess communication', pages 355 - 390 voir page 367, ligne 16 - page 389, ligne 6	1-8

1

RAPPORT DE RECHERCHE INTERNATIONALE

Renseignements relatifs aux membres de familles de brevets

Dem. : Internationale No
PCT/FR 93/01219

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
EP-A-0371229	06-06-90	US-A- 5124909	23-06-92
		JP-A- 2171951	03-07-90
GB-A-2176636	31-12-86	AUCUN	
US-A-5123094	16-06-92	AUCUN	

1/4

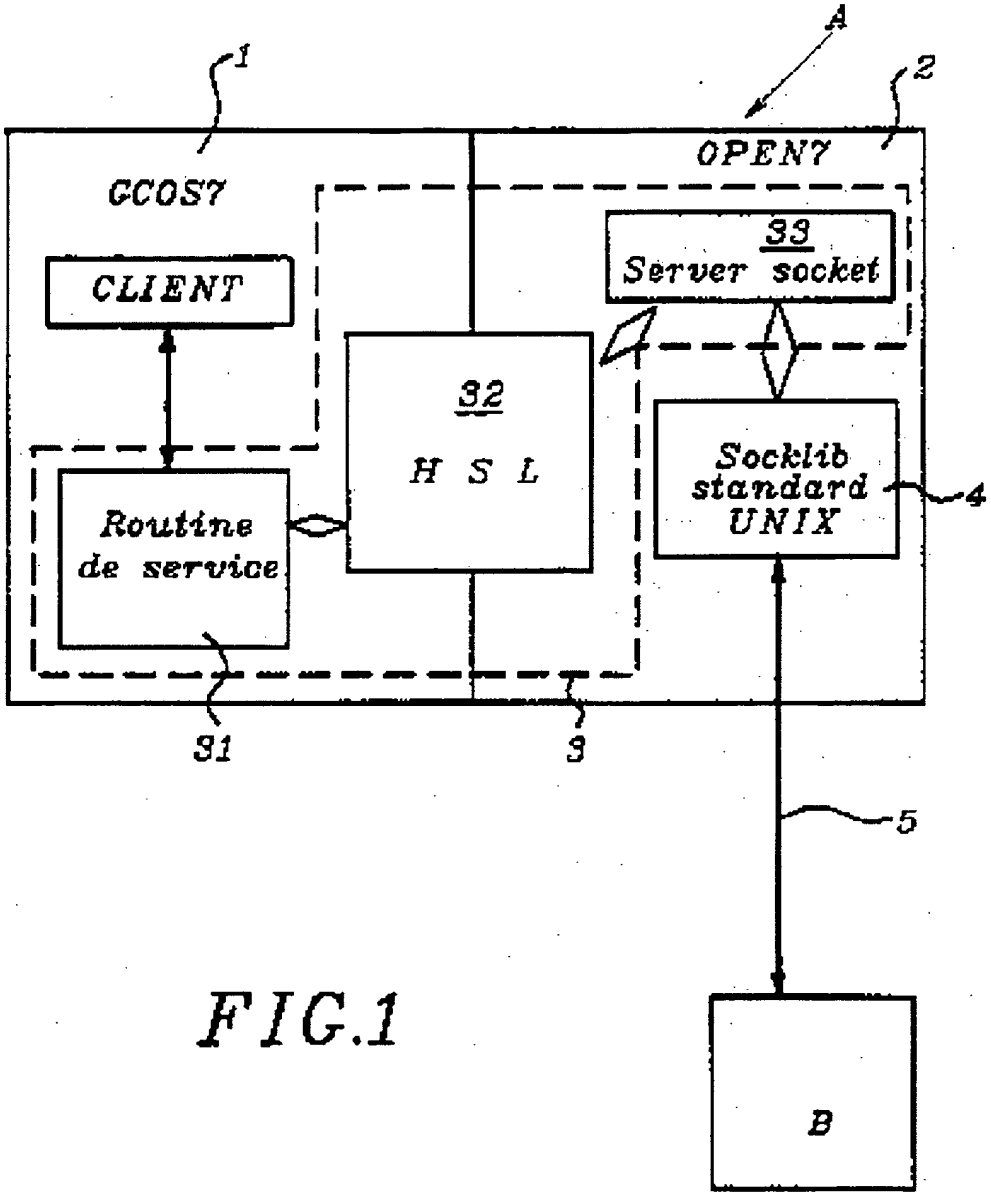


FIG.1

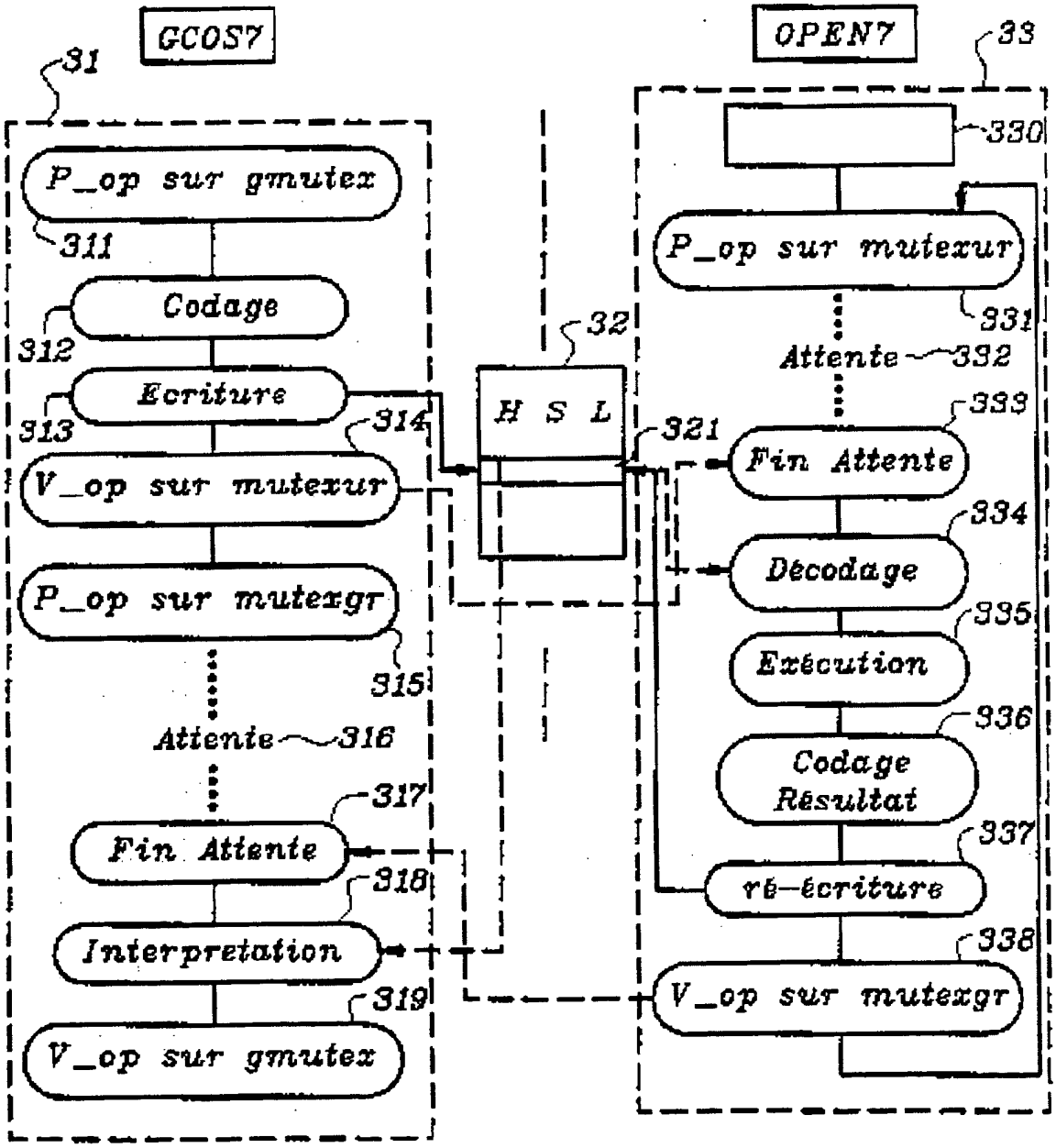


FIG.2

FIG.3

func_no (integer)	:	*une valeur qui represente la fonction*
jp (integer)	:	*le J–no et le P–no pour lesquels la demande est faite*
func_value (integer)	:	*la valeur de la fonction (remplie par socket_serv après l'execution de la fonction*
loc_errno (integer)	:	*0 si la fonction s'est executée normalement*
sys_ser_num (integer)	:	*pour identifier la machine qui fait la demande*
bkst0 (short)	:	*pour identifier la machine qui fait la demande*
rful (short)	:	*reserve*
int1,int2,int3, int4,int5,int6 (integer)	:	*ces champs servent a stocker les valeurs des param–etres dans le cas de parametres "integer"*
buf (char[2000])	:	*une zone de caracteres permettant de stocker les valeurs de parametres de type autre que "integer"*

321,322

func_no	*JP*	*func_value*	*loc_errno*	*sys_ser_num*	*bkst0*	*rful*	*int1*	*int2*
int3	*int4*	*int5*	*int6*	*buf*				

Function name	Fonction number
uuid_gen	1
fcntl	2
socket	3
bind	4
close	5
gethostbyaddr	6
gethostbyname	7
gethostname	8
getsockname	9
recvfrom	10
sendto	11
setsockopt	12
select	13
closeall	14
gettimeofday	15
recvmsg	16
sendmsg	17
accept	18
connect	19
ioctl	20
listen	21
getsockopt	22
getpeername	23
send	24
recv	25

FIG.4